CS 470 Group-8 Project
"Hospital Management System" Project Report

GitHub repository link: https://github.com/sajibdebnath/CS470_Group-8_Hospital_Management_System

YouTube link: https://youtu.be/J9F3-cnjE_A

Group Members:

1. **Name**: Sajib Debnath, **WIU ID**: 923-60-2769, **Email**: s-debnath@wiu.edu
2. **Name**: Md Zakir Hossain Zamil, **WIU ID**: 923-50-3436, **Email:** mzh-zamil@wiu.edu
3. **Name**: Clerique Ward, **WIU ID**: 923-663-392, **Email**: cts-ward@wiu.edu

1. Project Overview

The Hospital Management System (HMS) is a comprehensive and dynamic web-based platform developed using Flask (Python Web Framework) and Oracle Database Management System. The primary goal of this system is to provide an efficient, user-friendly, and scalable solution for managing key hospital operations, including patient information, doctor profiles, departmental organization, room occupancy, appointment scheduling, and billing records.

Hospitals and healthcare providers require effective systems to streamline their day-to-day administrative tasks. Manual processing often leads to delays, redundancy, and errors in critical patient records. The HMS addresses these issues by digitizing hospital records and automating interactions between patients, doctors, and administrators. It supports CRUD (Create, Read, Update, Delete) operations, ensuring real-time synchronization between the frontend GUI and the backend Oracle database.

This system is ideal for educational, training, and prototype healthcare system development purposes and can be further extended for commercial deployment with enhanced security, authentication, and patient engagement modules.

2. Technologies Used

The project combines modern web technologies with robust database integration to offer a seamless user experience:

- **Frontend**:
  - HTML5 and CSS3 for structural and stylistic markup
  - Bootstrap 5 for responsive design
  - Jinja2 templating engine for dynamic content rendering
- **Backend**:
  - Python 3.13
  - Flask 2.2 for routing and web service management
  - Flask-SQLAlchemy 3.0.3 as ORM (Object Relational Mapper)
- **Database**:
  - Oracle Database 19c
  - cx-Oracle 8.3.0 Python module for connectivity

- **Other Libraries**:
  - MarkupSafe, Click, Greenlet, Importlib-Metadata, Zipp, etc.

3. System Entities, attributes and Descriptions

Entities:
- **Patient**: (pno, pname, pgender, page, pward, pstatus, room_no), Represents individuals admitted or consulting at the hospital.
- **Doctor**: (dept_id, dept_name, floor, head), (dno, dname, specialization, contact, dept_id), Contains doctors' information and specializations.
- **Department**: Medical departments such as Cardiology, Neurology.
- **Room**: (room_no, room_type, is_occupied), Hospital rooms, types, and occupancy.
- **Appointment**: (appt_id, pno, dno, appt_date, appt_time, reason), Booking details linking patients and doctors.
- **Billing**: (bill_id, pno, amount, payment_method, paid_status), Financial records related to patient care.

Each entity is implemented with SQLAlchemy classes and database tables.

4. Entity-Relationship (ER) Diagram

The ER diagram presents a conceptual model of how different entities are interrelated:
- One patient can have many appointments.
- Each appointment is linked to one doctor.
- A doctor belongs to a single department.
- A patient may be assigned to a room.
- Each patient may have multiple billing records.

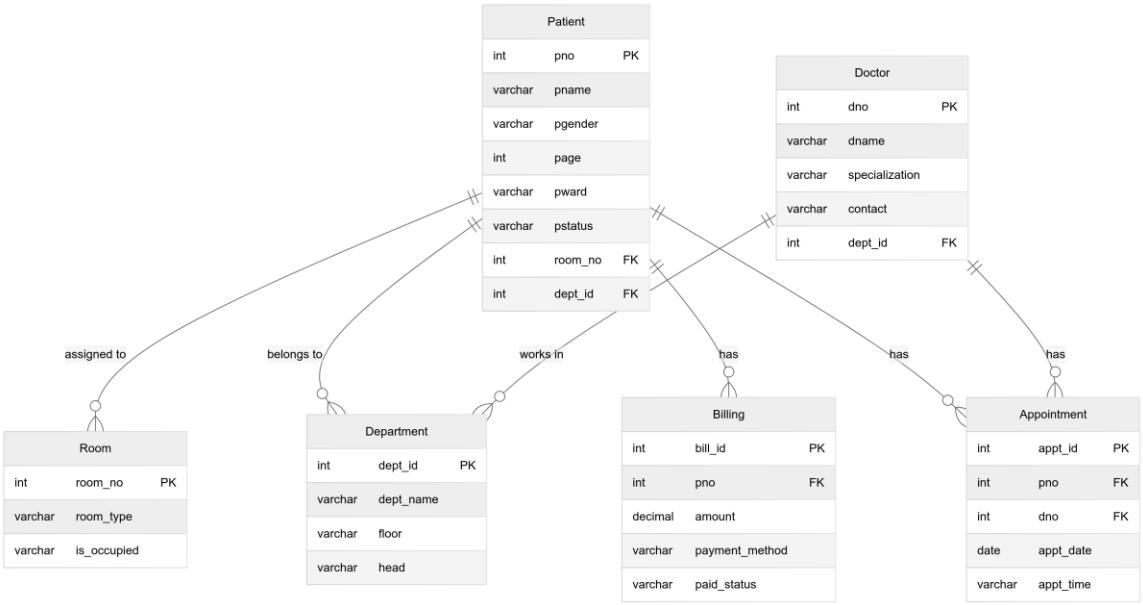

Figure 1: Hospital Management System (HMS) ERD for visual overview.

5. Database Schema and Constraints

The SQL schema includes all necessary CREATE TABLE statements along with primary and foreign key constraints. It ensures referential integrity among tables such as Patient, Doctor, Department, Room, Appointment, and Billing.

All database setup SQL is provided in **SQL G8.sql**

6. Application Modules and Flask Routes

The application is modularized using Flask blueprints and has routes defined for each entity:
- /: Homepage
- /patient: Patient management
- /doctor: Doctor management
- /department: Department management
- /room: Room management
- /appointment: Schedule appointments
- /billing: Manage billing records

The logic for handling form submission and database interaction resides in insert.py.

7. Data Interaction and ORM Mapping

SQLAlchemy ORM maps Python classes to Oracle SQL tables. Models include class definitions for Patient, Doctor, Department, Room, Appointment, and Billing. These classes include field types, constraints, and relationships.

8. GUI Design and User Experience

User interface is designed with Bootstrap and rendered using Jinja2. Pages such as billing_edit.html use styled forms for better user interaction.

Screenshots:
- Homepage UI:

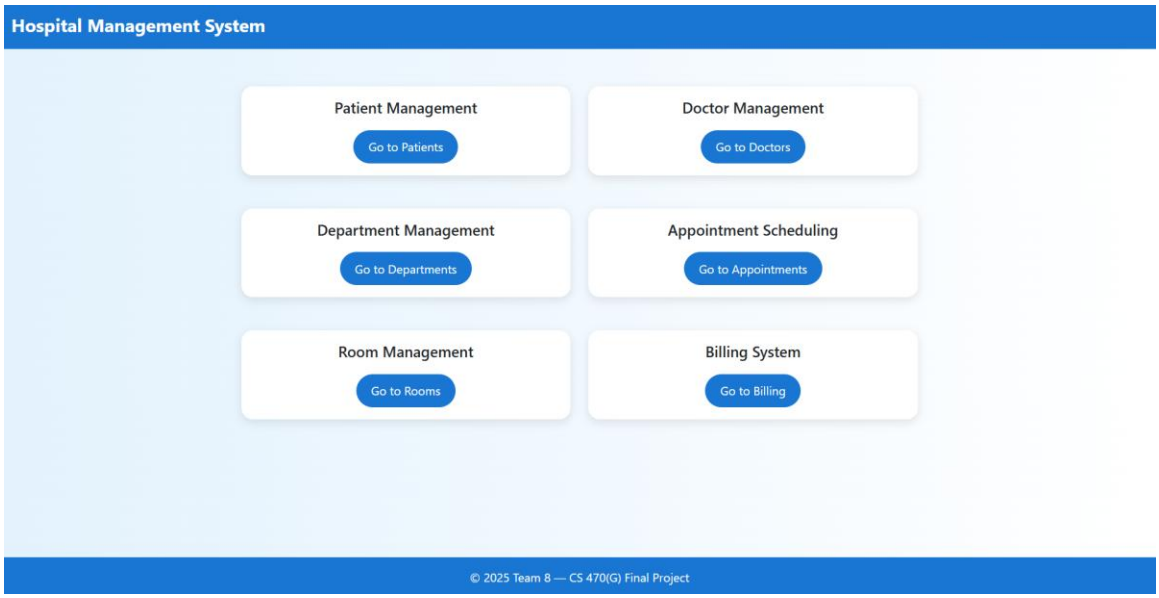

Figure 2: Hospital Management System (HMS) Home page.

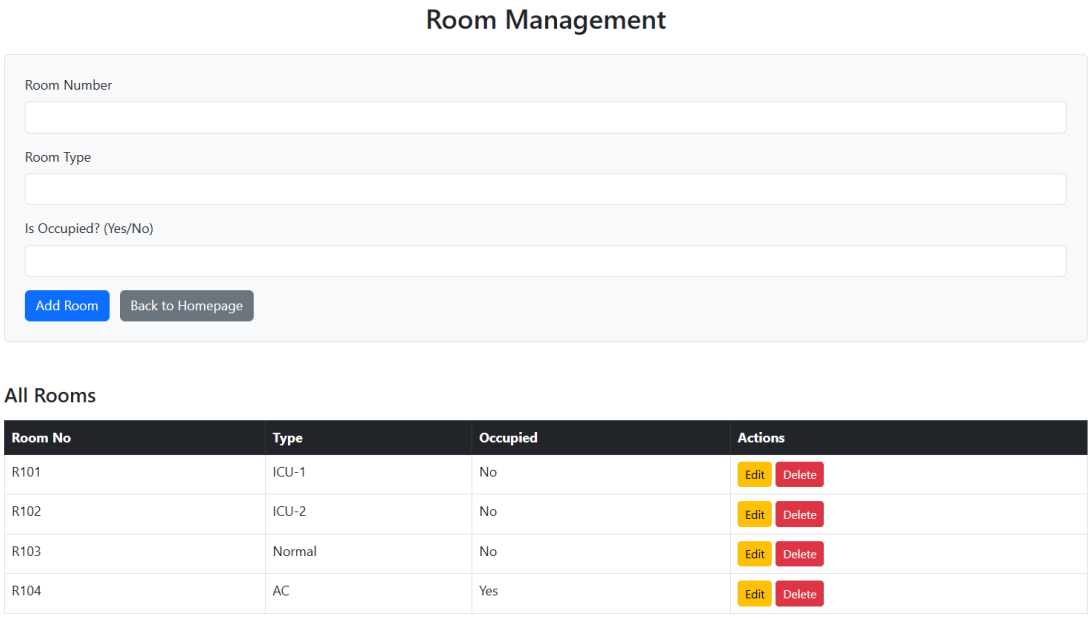

Figure 3: Room Management System Add Room and view data.

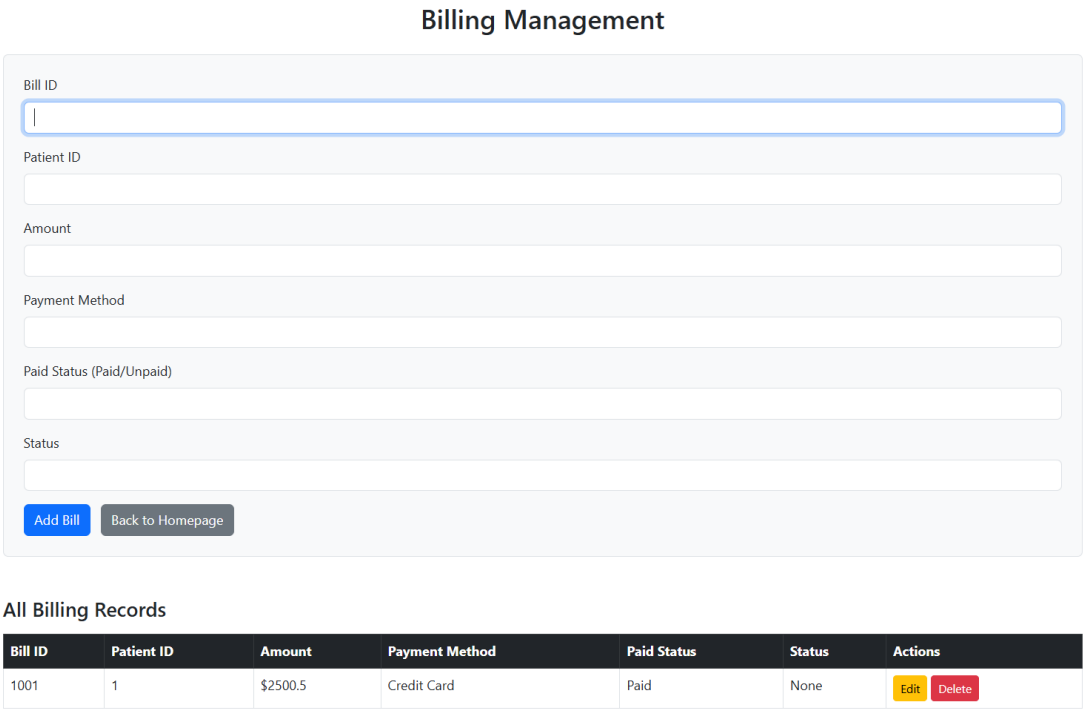

Figure 4: Billing Management add bill, view, edit and delete data.

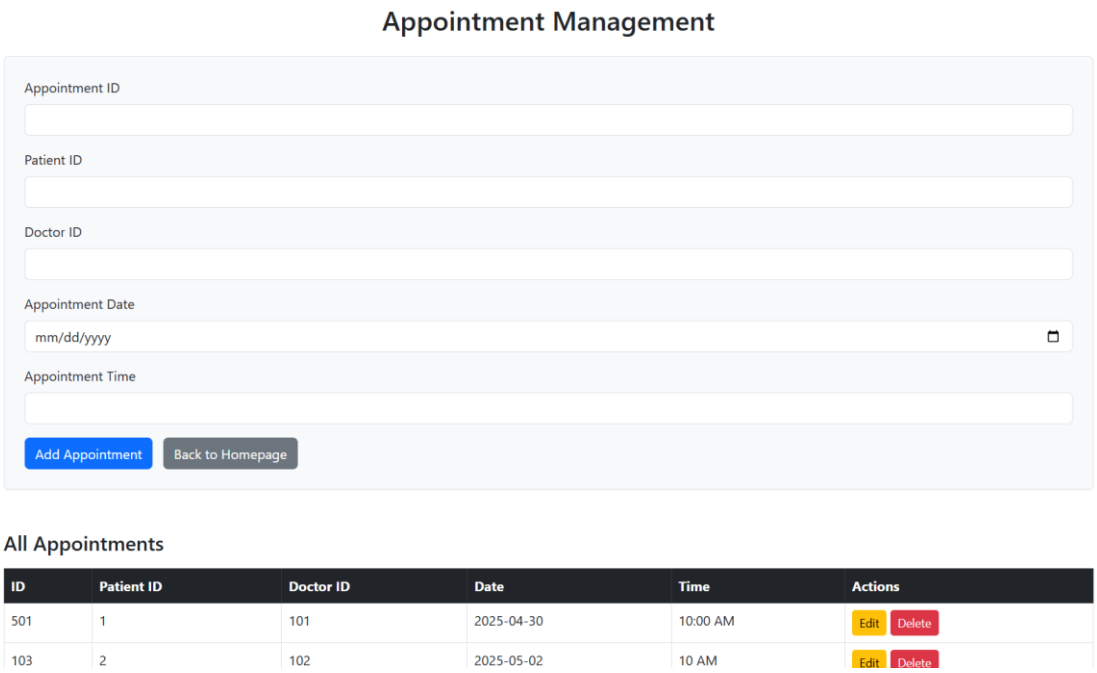

Figure 5: Appointment Management Add Appointment view, edit and delete data.

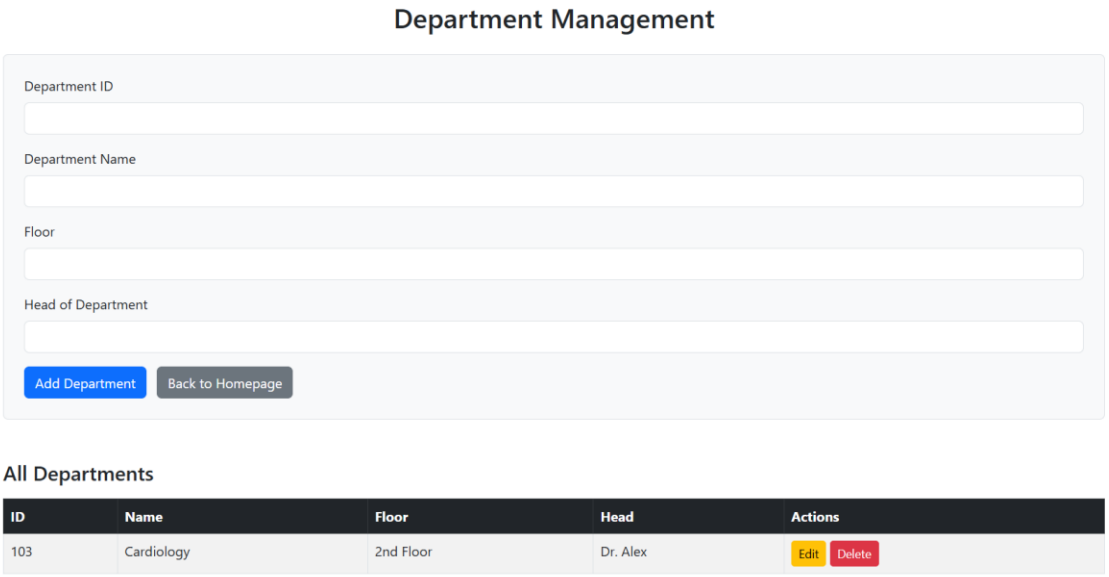

Figure 6: Department Management Add Department view, edit and delete data.

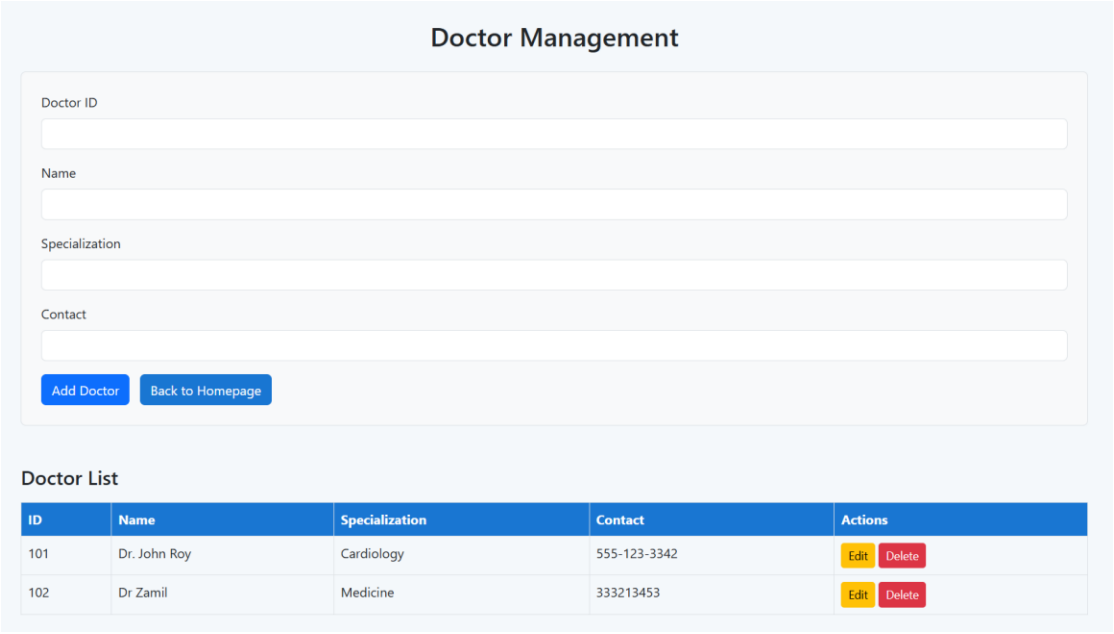

Figure 7: Doctor Management Add doctor view, edit and delete data.

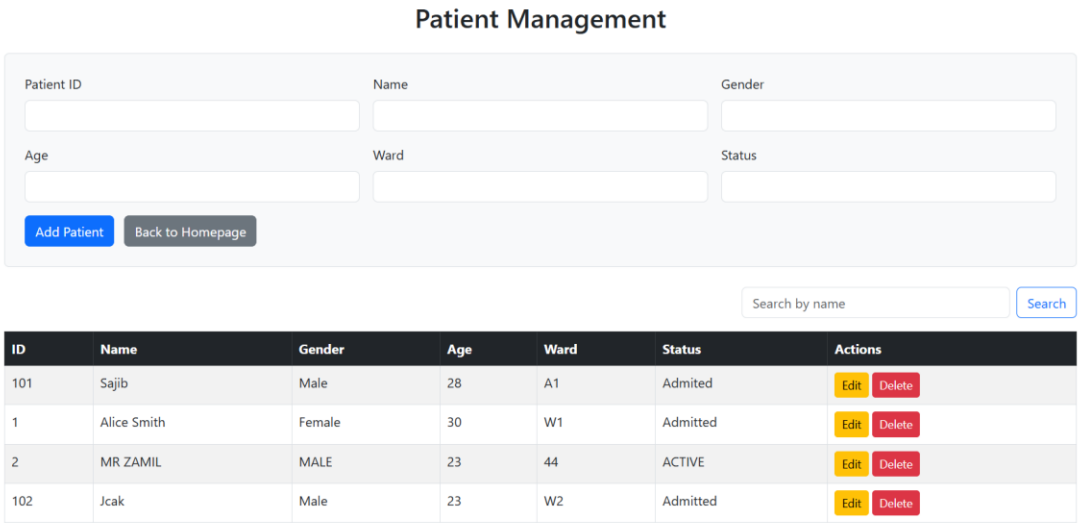

Figure 8: Patient Management Add patient view, edit and delete data.

Figure 9: Edit Doctor Form

9. Installation and Execution Guide

Steps:
1. Setup Oracle DB and execute **SQL G8.sql**
2. Install Python dependencies:
   pip install -r requirements.txt
3. Run the application:
   python app.py
4. Access via: http://127.0.0.1:5000

10. Strengths and Future Scope

Strengths:
- Oracle DB integration
- Modular Flask design
- Bootstrap frontend
- Scalable and maintainable structure

Future Scope:
- User authentication (admin/patient)
- Analytics dashboard
- Report exports (PDF/CSV)
- Email notifications

11. Conclusion

This project demonstrates the integration of Python Flask with Oracle DB for an efficient hospital management workflow. It provides a functional prototype with scalable architecture, user-friendly interface, and modular code structure. Future enhancements can make it a production-ready enterprise application.